

SOPHAIL: A CRITICAL ANALYSIS OF SOPHOS ANTIVIRUS

INTRODUCTION

- » Anti-virus vendors often assert they must be protected from scrutiny and criticism.
- » I firmly disagree, if close examination of a security system weakens it, then the system must be flawed.
- » Obscurity removes the incentive to improve and neuters our most effective defence: peer review.
- » This talk presents the results of an investigation into Sophos Anti-virus, and an analysis of the product claims made by the vendor.

ABOUT ME

- » My research is primarily vulnerability discovery, exploitation and mitigation.
- » My main focus is on operating system and low level security, primarily memory corruption related.
- » Anti-virus is interesting to me as a large exposed attack surface, not a mitigation.
- » This talk will be from the perspective of a vulnerability researcher.
- » Don't worry - this isn't a sales pitch 😊

MALWARE AND ANTIVIRUS

- » There is little intersection between my work and the work of AV vendors, as malware is a post-compromise concept.
 - An attacker must already be able to execute code (typically access to a user who makes poor trust decisions).
 - It's true that malware could be distributed as the payload for an exploit, but we would respond differently.
- » Most security researchers work on the assumption that users make good trust decisions, and we find ways to subvert that.
- » Antivirus vendors work on the assumption that users cannot make good trust decisions, and try to compensate.

TRUST DECISIONS

- » Unfortunately, the anti-virus vendors are right:
 - Many users (perhaps the majority) are simply unwilling or incapable of making good trust decisions.
 - Maybe they shouldn't have to, it can be complex and confusing.
- » We both agree that the solution to this problem is to offload the decision to someone who is capable.
- » Where we diverge is how to implement that process.

ANTIVIRUS

- » The solution proposed and implemented by anti-virus vendors is essentially a complex blacklisting scheme.
- » People understand that anti-virus is imperfect and reactionary in nature, but generally believe it's better than nothing.
 - Of course, there is an infinite set of possible malware samples; blacklisting will always be imperfect.
 - But perhaps this is reasonable, how do you evaluate that?
- » The same way you should evaluate any security product.

“Can this additional attack surface reduce my overall exposure?”

SECURITY PRODUCTS

- » We don't know how to build perfectly secure systems, because all software has a non-zero defect rate.
- » The best that we can do is raise the cost of a compromise above it's potential value.
 - Security products can be a legitimate part of that.
- » There are two classes of (legitimate) security products
 - Those that reduce overall attack surface (e.g. Firewall).
 - Those that increase attack surface in exchange for strengthening, enforcing or auditing some weak security boundary (e.g. Two factor authentication scheme).
- » Anti-virus falls in to the second category.

EVALUATION

- » Anti-virus promises to reduce the need for users to make trust decisions in exchange for a dramatic increase in exposed attack surface.
- » If your users are permitted to make trust decisions but cannot do so safely, maybe that is a good trade, as you need to do something about that urgently.
- » We need to be able to evaluate and understand what AV does to evaluate that.
 - How much attack surface are we trading?
 - How effective is the solution?
 - Does the vendor understand the problem?

MARKETING

- » Unfortunately, vendors will generally not tell you what it is they do, or how they verify the efficacy of the techniques they use.
- » Antivirus vendors do not publish technical specifications, the closest we can get promotional and marketing material.
 - Usually high-level doublespeak with little technical substance.
 - Mostly vague pseudo-scientific claims about “threats”.
 - How can we evaluate their claims?

EVALUATION

- » This talk is intended to help provide some of the missing technical specifications for Sophos Antivirus.
- » It's hoped that this will help those tasked with evaluating an AV product to do so thoroughly.
- » With a better understanding of what these products do, you can make more informed decisions about whether they make sense.
- » More details will be available in the accompanying paper.

INDUSTRY TESTING

- » Industry antivirus comparisons use variations of the same methodology in their “labs”.
 - Essentially they have a corpus of collected malware, and then they scan it and record results.
 - Little understanding of what the product they’re testing does, and treat it entirely as a blackbox.
 - This is analogous to reviewing cryptosystems based on how random the output looks!
- » The security press tend to focus on usability and perception, or at best anecdotal ad-hoc results.
 - Generally do not have the expertise to effectively evaluate vendor claims, and have to take them at their word.

ANALYSIS

- » This talk is an attempt to remedy this.
- » I've reverse engineered the core Sophos Antivirus engine and can provide some of the missing specifications.
- » Only reverse engineering techniques and tools that are publically available and easily accessible to attackers.
 - I have no access to proprietary knowledge.
 - I'm not discussing vulnerabilities, only design and implementation analysis.

ABOUT SOPHOS

- » Sophos is a widely used antivirus implementation, which often scores well in industry tests.
 - Ship a number of security related products.
 - Only discussing the Antivirus product.
- » The version tested was the latest available at the time of writing.
- » I've shared some drafts with Sophos before publication, they were good natured about it, and receptive to criticism.

Sophail

SIGNATURE MATCHING

SIGNATURE MATCHING

“Sophos' Dynamic Code Analysis technology utilizes sophisticated pattern matching techniques and identifies viruses by rapidly analyzing specific code sequences known to be present within a virus. Virus patterns are created to ensure that the engine catches not only the original virus but derivatives within the same virus family.”

- » Static file signatures are the core mechanism Sophos uses to blacklist known malicious code.
- » I've reverse engineered the core Sophos signature matching VM, and the Sophos signature file format
- » I've also created tools to allow inspection of the signatures, and to allow interested readers to reproduce my results.
- » Source code will be made available along with an accompanying paper shortly.

SIGNATURE FILE FORMAT

- » All Sophos signature files are distributed in a container format called “Sophtainers”.
- » The Sophtainers contain subsections called partitions that can be extracted as appropriate, each partition begins with a 32bit identifier describing the contents.
- » Sophtainer files begin with a mandatory Table Of Contents section which describes the location of the Partition List and the Section List.
 - Sections can be compressed and/or encrypted, however the only encryption algorithm supported is XOR, and the key is included in the file.
 - Sophos statically link a very old zlib (version 1.1.3) for compression.
- » All Sophos files use this same container format.

SOPHTAINER HEADER

```
0000000: 534f 5048 0100 7c92 0d3a e901 9c41 b49c SOPH..|:....A..
0000010: 639f bd14 a3bb 0100 544f 4300 5400 0000 c.....TOC.T...
0000020: 0786 cd0f 504c 4953 1000 0000 0100 0000 ....PLIS.....
0000030: 2ffd 0500 534c 4953 3800 0000 0100 0000 /...SLIS8.....
0000040: 7664 6c30 3100 0000 b3fc 0500 6c00 0000 vdl01.....l...
0000050: 434f 5a4c 4352 0000 0000 0000 4348 0001 COZLCR.....CH..
0000060: 0400 0000 1d0a a5e8 83a4 0a00 5345 4354 .....SECT
0000070: 7664 6c30 3100 0000 b3fc 0500 78da 8c7d vdl01.....x..}
0000080: 0758 53d7 fb7f 2010 f610 f70a d73d 8364 .XS... ..=.d
0000090: 0727 214c 0544 41dc 2340 8408 2436 0414 .!L.DA.#@..$6..
```

SIGNATURE FILE FORMAT

- » Signature files are signed by Sophos, however they use a proprietary authentication scheme invented by Sophos.
 - I've prepared a thorough examination of the authentication scheme later in this presentation.
- » The virus signatures are contained within the 'IDE' section of sophtainer files, which contain variable width chunks containing details about the signature and the signature themselves.

DECIPHERING A SIGNATURE

- » Each signature definition contains the Virus Name, followed by one or more bytecode programs that describe the file.
- » The bytecode program is executed on a proprietary VM for each input, deciding if the contents matches.
- » I've written tools to extract and disassemble the bytecode, so it's possible to examine the bytecode in more detail.

BYTECODE VIRTUAL MACHINE

- » The VM is a simple stack-based machine with single byte opcodes, and variable width operands.
 - RPN-like stack for computation.

Opcode		Description
VDL_OP_CRC32	96	Match crc32 n bytes (ones complement).
VDL_OP_NEXT	FA	Increment the file pointer.
VDL_OP_READSW	E1	Read word onto stack.
VDL_OP_LOADIW SW	DE	Load immediate word onto stack.
VDL_OP_SEEKSW	E8	Pop word, seek to absolute offset.
VDL_OP_SEEKIB	EB	Move file pointer forward n bytes.
VDL_OP_FADJUST SW	CB	Adjust next value on stack.
VDL_OP_SUBSW	D6	Pop two words, subtract, push result.
VDL_OP_SEEKIW	F8	Seek to immediate offset.

SAMPLE BYTECODE PROGRAM

Sample signature for “Attention 629”

0000: fb eb 7b	literalw	eb 70
0003: fc 90	literalb	90
0005: eb 03	seekib	03
0007: 96 06 2c b0 28 73	crc32	06 2c b0 28 73
000d: fa	next	
000e: fa	next	
000f: fb 75 02	literalw	75 02
0012: eb 09	seekib	09
0014: 96 27 13 e1 98 0e	crc32	27 13 e1 98 0e
001a: ed	hlt	

BYTECODE VIRTUAL MACHINE

- » This sample is above average complexity, includes some file pointer manipulation.
- » Most signatures follow a simple pattern:
 1. Identify a small literal byte pattern.
 2. CRC32 the following data for n bytes.
 3. CRC32 the data following that for n bytes.
- » Possibly automatically generated, as they rarely make sense.
 - Often irrelevant or dead code sequences.
 - Often very small code sequences (five or six bytes).

SIGNATURE MATCHING

- » The core of most signatures I've examined has been CRC32 some byte sequence.
- » The rationale for relying on such a weak scheme is unclear.
 - Signature schemes with far superior properties have been published.
 - Research in this area suggests CFG matching is the best known solution.
- » Such heavy reliance on non-cryptographic hashing introduces a number of problems.

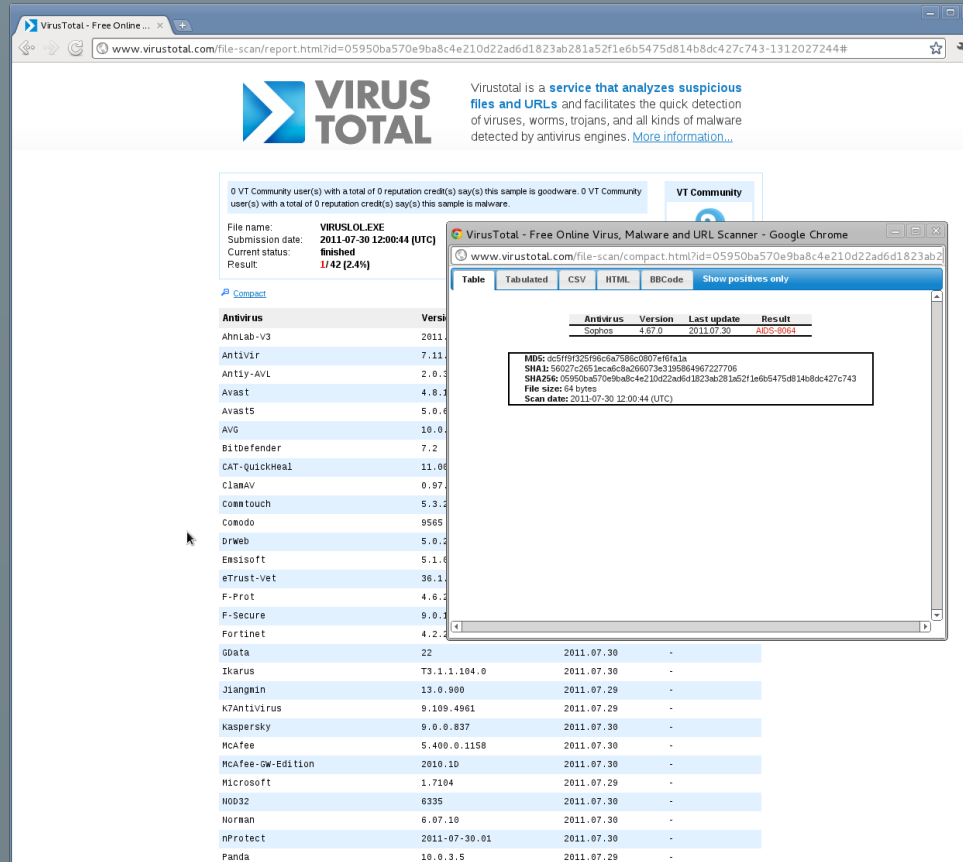
COLLISION RESISTANCE

- » It is self-evident that one of the core goals of an antivirus signature scheme should be to minimise false positives.
- » This is a popular research topic, with a large body of relevant work.
 - Sophos have ignored this, resulting in a very weak scheme.
 - In fact, not only is it obvious that collisions are widespread.
 - We can also automatically generate pre-images for many Sophos signatures. I've written the code to do this.
 - Pool pollution attacks.

GENERATING PRE-IMAGES

- » Generating CRC32 pre-images is trivial.
 - Even if it wasn't, even a naïve attacker can brute force 32bits in minutes on commodity hardware.

SCANNING PRE-IMAGES



VirusTotal - Free Online Virus, Malware and URL Scanner - Google Chrome

www.virustotal.com/file-scan/report.html?id=05950ba570e9ba8c4e210d22ad6d1823ab281a52f1e6b5475d814b8dc427c743-1312027244#

VIRUS TOTAL

VirusTotal is a **service that analyzes suspicious files and URLs** and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware detected by antivirus engines. [More information...](#)

0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is goodware. 0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is malware.

File name: **VIRUSOLEXE**
Submission date: **2011-07-30 12:00:44 (UTC)**
Current status: **finished**
Result: **1/42 (2.4%)**

Antivirus

Antivirus	Version	Last update	Result
AhnLab-V3	2011.	2011.07.30	-
AntVir	7.11.	2011.07.30	-
Antiy-AVL	2.0.3.	2011.07.30	-
Avast	4.8.3.	2011.07.30	-
Avast5	5.0.6.	2011.07.30	-
AVG	10.0.	2011.07.30	-
BitDefender	7.2.	2011.07.30	-
CAT-QuickHeal	11.00.	2011.07.30	-
ClamAV	0.97.	2011.07.30	-
Comodo	5.3.2.	2011.07.30	-
Comodo	9565.	2011.07.30	-
DrWeb	5.0.2.	2011.07.30	-
Emsisoft	5.1.4.	2011.07.30	-
eTrust-Vet	36.1.	2011.07.30	-
F-Prot	4.6.2.	2011.07.30	-
F-Secure	9.0.3.	2011.07.30	-
Fortinet	4.2.2.	2011.07.30	-
GData	22.	2011.07.30	-
Ikarus	73.1.1.104.0	2011.07.30	-
Jiangmin	13.0.900	2011.07.29	-
K7AntiVirus	9.109.4961	2011.07.29	-
Kaspersky	9.0.8.837	2011.07.30	-
McAfee	5.400.0.1158	2011.07.30	-
McAfee-GW-Edition	2010.10	2011.07.30	-
Microsoft	1.7104	2011.07.29	-
NOD32	6335	2011.07.30	-
Noran	6.07.10	2011.07.30	-
nProtect	2011-07-30.01	2011.07.30	-
Panda	10.0.3.5	2011.07.29	-

Table | **Tabulated** | **CSV** | **HTML** | **BBCode** | **Show positives only**

Antivirus	Version	Last update	Result
Sophos	4.67.0	2011.07.30	ADG-8064

MD5: dc5f9f325f96c6a7586c0807e6fa1a
SHA1: 5607c2951ec4c8a298077a3186844967227706
SHA256: 05950ba570e9ba8c4e210d22ad6d1823ab281a52f1e6b5475d814b8dc427c743
File size: 64 bytes
Scan date: 2011-07-30 12:00:44 (UTC)

SIGNATURE QUALITY

- » Sophos claim that their researchers spend time creating good quality generic signatures.
- » I tested this claim by examining a random sample of signatures.
 - They have tens of thousands of signatures, I cannot be exhaustive.
- » Frankly, I see little evidence they're aware of the context of the signatures they're creating.
- » Many signatures are obviously automatically generated.
- » Others checksum dead, trivial or irrelevant code.
- » There are some counter-examples, but they are the minority.

SUMMARY

- » Sophos signatures are bytecode for a proprietary VM.
- » The signatures rely heavily on very weak CRC32.
- » Even non-automatically generated signatures tend to be of poor quality, often matching irrelevant or dead code sequences.
- » The Sophos signature scheme can be considered weak at best.

Sophail

EXPLOIT MITIGATION

EXPLOIT MITIGATION

- » Sophos sell a runtime exploit mitigation system for Windows systems.
- » They should be applauded for attempting to do so, however their system is poorly researched and badly broken.
- » “Defeating” the exploit mitigation is trivial, but not really necessary, as it does very little mitigation.

SOPHOS BUFFER OVERFLOW DETECTION

*"This detects
vulnerabilities"*

*security
and*

Host Intrusion Prevention... x

www.sophos.com/en-us/why-sophos/innovative-technology/hips/layers-of-detection.aspx

Login Register | About us Press Contact us Careers Global websites Search go

WHAT WE DO
Your peers | Why Sophos | Products | Support | Product Center | Security News | Events | Partners

Home > Why Sophos > Innovative Technology > HIPS > Layers of Detection

SOPHOS

Reputation
Our Products
Our People
Our Customers

Innovative Technology
Anonymizing Proxies
Antivirus
Application Control
Device Control
Data Loss Prevention
Encryption
HIPS
Layers of Detection
Link

Sophos HIPS: Four Layers of Detection

Effective protection fine-tuned by SophosLabs for you

HIPS detection, in four layers

Our threat detection engine analyzes the behavior of code before it executes and prevents it from running if it is considered to be suspicious or malicious. It also uses runtime detection to intercept threats.

Pre-execution detection

1. Behavioral Genotype® Protection: Tuned to detect variants, families (like the [Storm worm](#)) and large categories of malware (like encrypted malware), Genotype Protection guards against unknown malware by analyzing behavior before code executes. It uses pre-execution behavior, as SophosLabs experts do the fine tuning.

2. Suspicious behavior definitions

4. Buffer overflow detection: A buffer overflow attack is reported when an attempt is made to exploit a running process using buffer overflow techniques. This detection system will catch attacks targeting security vulnerabilities in both operating system software and applications.

[Read more about buffer overflow detection](#)

4. Buffer overflow detection: A buffer overflow attack is reported when an attempt is made to exploit a running process using buffer overflow techniques. This detection system will catch attacks targeting security vulnerabilities in both operating system software and applications.

[Read more about buffer overflow detection](#)

[Try Sophos products now](#)
Download now

Learn more about our products

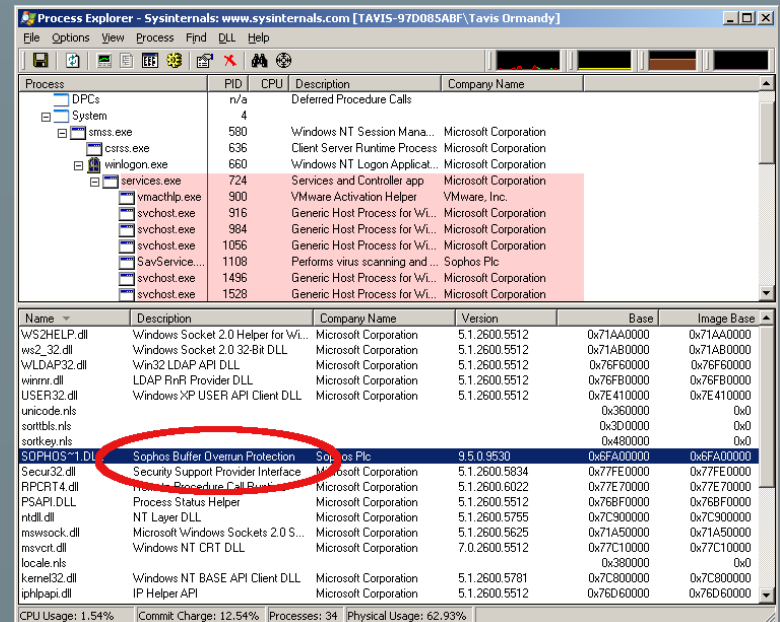
Customer quotes
"I like the fact that I can call the same vendor for every issue. The more we bundled, the better the pricing got, too"
Network Admin, Taco Bueno

SOPHOS BUFFER OVERFLOW DETECTION

- » The Sophos product is called “BOPS” for Buffer Overflow Protection Service.
- » The product is touted by Sophos as one of their four “Layers of Detection”.
- » Zero technical documentation or specifications available from Sophos, making it difficult to evaluate their claims.
- » Furthermore, the system was (naively) obfuscated to make it harder to understand, but posed little challenge to reverse engineer.

EXPLOIT MITIGATION DESIGN

- » BOPS is Implemented entirely in userspace.
- » Loaded into the address space of (most) processes at runtime using Appinit_Dlls.
- » Written using Microsoft Detours Professional, which is used to for API interception.



Process Explorer - Sysinternals: www.sysinternals.com [TAVIS-97D085ABF,Tavis Ormandy]

Process	PID	CPU	Description	Company Name
DPCs	n/a		Deferred Procedure Calls	
System	4			
smss.exe	580		Windows NT Session Mana...	Microsoft Corporation
csrss.exe	636		Client Server Runtime Process	Microsoft Corporation
winlogon.exe	660		Windows NT Logon Applicat...	Microsoft Corporation
services.exe	724		Services and Controller app	Microsoft Corporation
vmacthlp.exe	900		VMware Activation Helper	VMware, Inc.
svchost.exe	916		Generic Host Process for Wl...	Microsoft Corporation
svchost.exe	984		Generic Host Process for Wl...	Microsoft Corporation
svchost.exe	1056		Generic Host Process for Wl...	Microsoft Corporation
SavService...	1108		Performs virus scanning and ...	Sophos Plc
svchost.exe	1436		Generic Host Process for Wl...	Microsoft Corporation
svchost.exe	1528		Generic Host Process for Wl...	Microsoft Corporation

Name	Description	Company Name	Version	Base	Image Base
WS2HELP.dll	Windows Socket 2.0 Helper for Wl...	Microsoft Corporation	5.1.2600.5512	0x71AA0000	0x71AA0000
ws2_32.dll	Windows Socket 2.0 32-Bit DLL	Microsoft Corporation	5.1.2600.5512	0x71AB0000	0x71AB0000
WLDAP32.dll	Win32 LDAP API DLL	Microsoft Corporation	5.1.2600.5512	0x76F60000	0x76F60000
winmr.dll	LDAP RnR Provider DLL	Microsoft Corporation	5.1.2600.5512	0x76F80000	0x76F80000
USER32.dll	Windows XP USER API Client DLL	Microsoft Corporation	5.1.2600.5512	0x7E410000	0x7E410000
unicode.nls				0x360000	0x0
sorttbls.nls				0x3D0000	0x0
sortkey.nls				0x480000	0x0
SOPHOS-1.DLL	Sophos Buffer Overrun Protection	Sophos Plc	9.5.0.9530	0x8FA00000	0x8FA00000
Secur32.dll	Security Support Provider Interface	Microsoft Corporation	5.1.2600.5634	0x77FE0000	0x77FE0000
RPCRT4.dll	Remote Procedure Call Runtime	Microsoft Corporation	5.1.2600.6022	0x77E70000	0x77E70000
PSAPI.DLL	Process Status Helper	Microsoft Corporation	5.1.2600.5512	0x76F00000	0x76F00000
ntdll.dll	NT Layer DLL	Microsoft Corporation	5.1.2600.5755	0x7C900000	0x7C900000
mswsock.dll	Microsoft Windows Sockets 2.0 S...	Microsoft Corporation	5.1.2600.5625	0x71A50000	0x71A50000
msvcrt.dll	Windows NT CRT DLL	Microsoft Corporation	7.0.2600.5512	0x77C10000	0x77C10000
locale.nls				0x380000	0x0
kernel32.dll	Windows NT BASE API Client DLL	Microsoft Corporation	5.1.2600.5781	0x7C800000	0x7C800000
iphlpapi.dll	IP Helper API	Microsoft Corporation	5.1.2600.5512	0x76D60000	0x76D60000

CPU Usage: 1.54% | Commit Charge: 12.54% | Processes: 34 | Physical Usage: 62.93%

EXPLOIT MITIGATION DESIGN

- » Despite misleading claims to the contrary, BOPS only works on versions of Windows prior to Vista.
 - However, it is loaded on all versions of windows.
 - If version is Vista or higher, just doesn't do anything.
- » Sophos intended to implement two weak forms of runtime exploit mitigation.
 - Simple SEH Overwrite Protection.
 - Ret2libc detection and prevention.
- » Lets recap on the attacks they're trying to prevent...

SEH OVERWRITE PROTECTION

- » Traditionally a very simple method of exploiting stack buffer overflows on Windows.
- » Adoption of toolchain and runtime integrity verification from Microsoft has effectively neutered the technique.
- » However,
 - SafeSEH has a number of drawbacks.
 - SEHOP is only available in Windows versions since vista.
- » It's not entirely unreasonable to sell a product that implements SEHOP mitigation for XP (although there are a number of good quality free implementations).

EXCEPTION CHAIN VERIFICATION

- » SEHOP is essentially a solved problem. Matt Miller has published the most work on this subject.
- » The solution is to insert a canary at the tail of the exception handler chain, allowing the list integrity to be checked by the system at exception dispatch.
- » The system can verify the chain has not been tampered with before using it.
- » There's a good quality implementation available with source code from WehnTrust.

EXCEPTION CHAIN VERIFICATION

```
ChainCorrupt = TRUE;
while (CurrentRecord != 0xffffffff) {
    if (IsValidAddress(CurrentRecord->Next))
        break;
    if (CurrentRecord->Next == ValidationFrame) {
        ChainCorrupt = FALSE;
        break;
    }
    CurrentRecord = CurrentRecord->Next;
}
if (ChainCorrupt == TRUE)
    ReportExploitationAttempt();
else
    CallOriginalKiUserExceptionDispatcher();
```

SOPHOS SEHOP IMPLEMENTATION

- » Sophos's implementation was clearly inspired by Matt's Work, but is far weaker.
- » Demonstrates a fundamental misunderstanding of the problem.
- » I've prepared some pseudo code for comparison to Matt's implementation.

EXCEPTION CHAIN VERIFICATION

```
CurrentRecord = Tib->ExceptionList;

for (i = 0; i < 2; i++) {
    if (IsBadReadPtr(CurrentRecord, Size)) {
        break;
    }
    if (CurrentRecord->Handler >= Tib->StackLimit
    && CurrentRecord->Handler <= Tib->StackBase) {
        SuspendCurrentThread();
    }
    if (CurrentRecord->Next == -1) {
        break;
    }

    CurrentRecord = CurrentRecord->Next;
}
```

SOPHOS SEHOP IMPLEMENTATION

Essentially, Sophos just verify that the handler for the first two exception records don't point within the current thread stack.



SOPHOS SEHOP IMPLEMENTATION

- » This mitigation is ridiculously weak.
- » Even a low-skilled attackers can trivially defeat this mitigation.
- » Sophos misunderstood published information on this topic, resulting in a broken implementation of what is essentially a solved problem.
- » The obvious attack against Sophos SEHOP is simply ret2libc, but Sophos do appear to have considered this.

RETURN TO LIBC

- » Originally developed by Solar Designer to demonstrate weaknesses in early stack buffer overflow mitigations.
- » Still an important technique, although the attack has been generalised over time and is now sometimes referred to as “ROP”.
- » A strong ASLR implementation is generally considered the best defence against ret2libc.
 - If an attacker cannot predict where the code he wants is located, he cannot return to it.
- » However, it is a reasonable observation that ASLR is not strong on all Windows platforms, particularly XP.

PROTECTED FUNCTIONS

- » Sophos solution to ret2libc exploitation called “Protected Functions”, and is part of the BOPS system.
- » At load time, parses an encrypted configuration file listing Windows APIs that Sophos believe are most likely to be used by attackers in their ret2libc payload.
- » These routines are hooked with Detours.
- » From their detours callback, they verify that the caller was from a recognised module and not within the current thread stack.

PROBLEMS WITH PROTECTED FUNCTIONS

- » This simply doesn't make any sense, Sophos didn't understand the problem correctly.
- » Sophos were confused because Solar Designer originally returned into an exported library function (`system`).
 - Solar did this because it was convenient.
 - There was no technical requirement to actually do so!
- » Modern exploits have made finding collections of useful code sequences a science.
 - These are often referred to as gadgets.
- » Attacker can simply piece together the code he wants from other locations or call the routine indirectly.
- » Or, you know, just return into the code immediately after the Hook 😊

PROBLEMS (CONTINUED)

- » Sophos were attempting to enumerate known bad (again).
- » This just doesn't make sense in the context of re-purposing trusted code by an attacker.
 - All trusted code becomes untrusted.
 - This solution simply doesn't work.
- » Sophos return-to-libc mitigation is useless, and I would recommend against it's use.

BOPS CONFIGURATION SYSTEM

- » Sophos keep the list of routines they hook a secret, and also a list of ImageNames they whitelist.
- » Their configuration file is encrypted with a hardcoded 64bit symmetric key.
 - Of course, this is trivially recoverable by attackers.
 - The key is 0xd6917912f2e43923 ;-)
- » However for an extra level of security, they then encrypt it using XOR, and the hardcoded 8bit key 0x93.
- » I've implemented their encryption scheme, called SPMAA, in C so that we can examine their mitigations.

BOPS CONFIGURATION

- You can use this command to extract the BOPS configuration file.

```
$ ./spmautil --output=results \
    --setkey=$( (0xd6917912f2e43923) ) \
    --filename=Config.bops \
    --decrypt
```

BOPS CONFIGURATION FILE

- » The BOPS library contains a list of ImageNames that it's “protections” apply to, some examples are
 - Quicktimeplayer
 - Acrord32
 - Outlook
 - Powerpnt, etc.
- » Even if BOPS mitigation was useful, they simply wouldn't apply to any other software.

Sophail

SPMAA ENCRYPTION

SPMAA ENCRYPTION

- » SPMAA is a proprietary 64bit feistel block cipher used throughout Sophos products.
- » It's undocumented, unpublished, and **not** peer reviewed.
- » Sophos rely on it almost exclusively for cryptography in their products.
 - Well, they also use XOR encryption a lot 😊
 - I've reverse engineered it and have a simple C implementation for review.
 - Used for authentication of IDE files.

SPMAA ENCRYPTION

- » Despite the inherent weaknesses in a 64bit cipher, it doesn't look too badly designed.
 - It hasn't been peer reviewed, but it isn't obviously weak.
 - Probably designed by a real cryptographer, but dated and obsolete now.
- » Unfortunately, it simply doesn't matter.
 - Sophos misuse it throughout their products.
 - It's used for authentication and secrecy incorrectly, Effectively reducing it to an obfuscation scheme.

SIGNATURE AUTHENTICATION

- » Sophos repurposed SPMAA to sign IDE files that users download.
 - 32bits of the 64bit internal state used.
 - This is not a real crypto mode, it's trivial to sign our own updates (but we could brute force 32bits trivially even if it wasn't).
- » This makes transport security essential, but Sophos simply do not use it.
- » They're relying on obfuscation to prevent active attackers from interfering with their product...This is not good.
- » I'm releasing tools to demonstrate this today.

SPMAA ENCRYPTION SUMMARY

- » Sophos use SPMAA to “encrypt” sensitive product data.
- » However SPMAA is a symmetric cipher, they simply hide the keys within the product
 - Sometimes XOR encryption is used as a second layer...?
- » It's trivial to recover the keys used using standard reverse engineering techniques.
- » Sophos use obfuscation where real cryptography could work, is well understood, and easily available to them.

CONCLUSION

- » Sophos do not use good quality encryption in their products.
- » The weak encryption they do use is misused, effectively weakening it to an obfuscation scheme.
- » Sophos try to augment their weak encryption with trivial XOR ciphers, obviously this doesn't work.
 - Sophos often rely on obfuscation for security.
 - Cryptography is outside their realm of expertise, they're often misusing the weak cryptographic primitives they do use.

REACTION

- » I spoke to Sophos about their use of SPMAA.
- » They conceded that their authentication scheme was weak.
 - They've assured me they will migrate to a real signature scheme in the next major versions, and are already working on testing it.
 - They explained that they will begin phasing out the use of SPMAA in their products.
- » I'm satisfied they understood the problem and are working to improve it.

Sophail

PRE-EXECUTION ANALYSIS

SOPHOS PRE-EXECUTION ANALYSIS

- » Sophos attempts to determine filetype characteristics that are used for matching by later stages.
- » Sophos attempt to identify container and archive file formats, and handle them appropriately.
- » For native executable code and Dynamic HTML and PDF documents, Sophos attempt to emulate or interpret the code.
 - Executable Packer support.
 - Archive extraction.

NATIVE CODE EMULATION

- » Sophos ship a simple x86 emulator that is used pre-execution on known executable types, and unknown file types.
 - Sophos calls unknown files `WTF` internally, and assumes they may be DOS COM files (reasonable).
- » The emulator executes the code for approximately 500 cycles on a simplistic machine (No CPL, x87, etc.).
- » They provide (very) simple DOS and NT stub.
- » Not a very good representation of x86, trivial to bypass or detect (just spin for 500 cycles).

NATIVE CODE EMULATION

- » Many features of the emulator are simply broken and have never worked.
- » The NT stub is a good example of Sophos misunderstanding fundamental NT concepts.
- » Sophos hardcoded the system call numbers for the emulator, not realising that these numbers vary across Windows releases, service packs, and updates.
- » They hardcoded the numbers they found in the SSDT from a Windows Server 2003 SP1 kernel, not realising this is not possible.

NATIVE CODE EMULATION

- » Similar problems exist throughout the emulator, and other Sophos subsystems.
- » The intention of the emulator is to record execution characteristics (mostly broken), and also to record memory references and allow simple decryption loops to run before using static signatures.
- » However even a naïve adversary can spin for 500 cycles before unpacking, so this is really of questionable value.

NATIVE UNPACKERS

- » Executable packers are widely used for software distribution, but are also a simple way for unskilled attackers to transform an executable into an equivalent but different executable.
 - Thus bypassing blacklisting schemes with zero skill.
 - Of course even a moderately skilled attacker would just write an equivalent program.
- » For this reason, many vendors (Sophos included) hype support for a large numbers of unpackers as a selling point.
- » Sophos do include support for a number of packed formats.

NATIVE UNPACKERS

Packer	Year	Summary
DIET	1992	Dr. Teddy's 'DIET' program for files.
PKLITE	1996	PKZIP for executable files.
LZEXE	1989	Fabrice Ballard's executable packer.
UPX	2001	The Ultimate Packer for eXecutables.
PETITE	1999	Ian Luck's executable packer.
ASPACK	1999	Alexey Solodovnikov's Packer.
FSG	2002	Fast Small Good , particularly popular in Poland.
PECompact	2001	PE Compact

NATIVE UNPACKERS

- » With the exception of PECompact (which appears to have been licensed from the vendor), these unpackers were written by Sophos.
- » They usually only support the default options and codecs, and are not tolerant of any modifications.
- » Testing the support was hard, as Sophos usually only support one very specific version.
 - I had to download dozens of versions from shareware archives to find working inputs.

NATIVE UNPACKERS

- » Based on the amount of trouble I had to go through to create executables that were supported...
 - Even naïve attackers would not have much trouble defeating this system.
 - Unpackers represent considerable attack surface for little gain, the unpackers supported are largely obsolete and irrelevant.

ARCHIVE AND CONTAINER SUPPORT

- » Sophos do support a large number of container and archive file formats.
- » The decoders vary in quality, some are simply bizarre and broken, or have bizarre exceptions.
 - For example, Sophos randomly exclude Siemens TriCore executables from the ELF decoder?!
 - Siemens/Infineon TriCore is used mostly for industrial microcontrollers.

GENES AND GENOTYPES

- » Sophos describe their “Genes and Genotypes” as a major selling point.
- » Undocumented, apart from marketing techno-babble.
- » A company (presumably) representing them patented it, so we can understand some of their rationale.
- » The idea is trivial, a Gene is just an arbitrary characteristic defined either statically in the core engine in subsequent identity updates.
- » Then they blacklist combination of these characteristics as malicious.

GENES

- » Some of these characteristics are defined during pre-execution analysis.
- » Genes can be characteristics found by emulation or at runtime (such as API calls), or by pattern matching (such as embedded strings).
 - Embedded URLs, IRC commands, Etc. are used heavily.

SOPHOS REACTION

- » Spoke to a Sophos engineer who seemed reasonable.
 - He explained that he felt antivirus had a place protecting non-technical users from untargeted attacks.
- » They read some early drafts, and explained that they will make some changes based on my criticisms.
 - Phasing out their use of SPMAA.
 - Using a real cryptographic scheme for signatures.
 - Re-evaluating BOPS.

CONCLUSION

- » Sophos demonstrate naivety in many areas key to the efficacy of their product.
 - Misusing Crypto, or relying XOR ciphers.
 - Misunderstanding NT System Calls.
 - Failing to comprehend rudimentary exploitation techniques.
- » Sophos technology is currently ill-equipped to deliver on the promises they make.

QUESTIONS?

Email: taviso@cmpxchg8b.com

Twitter: @taviso